Color Segmentation

Laboratory of Applied Intelligent Systems (AIS Lab)

author: Gilberto Decaro

Le due classi **AISYCrCbColorSeg** e **AISAiboYCrCbColorSeg** permettono di riconoscere fino a **32** classi di colori a partire da una immagine nello spazio colore **YCrCb** di dimensione arbitraria.
Le due classi si differenziano per il formato delle immagini che sono in grado di elaborare:

- **AISYCrCbColorSeg**: elabora immagini YCrCb intese come un'array di pixel, in cui ogni pixel è composto da tre byte: un byte y, un byte cr, un byte cb
- **AISAiboYCrCbColorSeg**: elabora immagini YCrCb in formato RAW, come ritornate dagli Aibo modello ERS7.

L'immagine ritornata dopo l'elaborazione è una matrice di interi senza segno delle stesse dimensioni in pixel dell'immagine originale, in cui l'i-esimo bit identifica l'i-esimo colore, se = 0 allora quel pixel non era del colore voluto, altrimenti è stato riconosciuto come appartenente al colore.

Il tempo di calcolo, dato n il numero di colori da riconoscere, è O(1); su un Aibo modello ERS7 il tempo di elaborazione è di circa 2msec a immagine.

Di seguito un esempio di riconoscimento dei colori, applicato sul layer H (208x160):

Illustration 1: Immagine originale

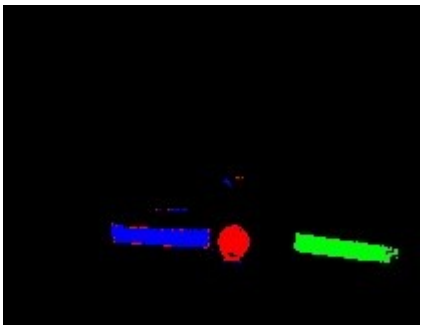

Illustration 2: Colori riconosciuti

Nell'immagine elaborata è presente un po' di “sporco” inteso come buchi negli oggetti (vedi palla rossa) e aloni mal riconosciuti (vedi pixel rossi attorno alla banda blu), problemi che possono essere risolti con una migliore taratura dei colori e con l'applicazione di filtri morfologici per la chiusura dei buchi (close).

Vantaggi rispetto CDT degli Aibo

Rispetto al Color Detection Hardware presente negli Aibo le classi AISAiboYCrCbColorSeg e AISYCrCbColorSeg hanno i seguenti vantaggi:

- Permettono di riconoscere fino a **32** colori (contro i 7 del CDT).
- Permettono una **maggiore granularità** nella specifica dei colori (255 valori di y contro 32 del CDT).
- Permettono **risoluzioni maggiori**. Il CDT degli Aibo ritorna immagini di

dimensioni 104x80, le due classi permettono di elaborare immagini di dimensioni arbitrarie.

Lo svantaggio delle due classi rispetto al CDT degli Aibo è ovviamente il tempo di calcolo. Il CDT è implementato in hardware nella telecamera, quindi la segmentazione non consuma risorse di sistema, obiettivo “difficile” da raggiungere senza modificare l'hardware degli Aibo.

Descrizione metodi

In entrambe le classi sono presenti i seguenti metodi di interesse per il riconoscimento dei colori:

- `bool` **`setColor`**`(Colors selectedLayer,`
  `unsigned char y,`
  `unsigned char crMin, unsigned char crMax,`
  `unsigned char cbMin, unsigned char cbMax):`
  Imposta un colore, in modo analogo ai metodi di open-R.
  - **selectedLayer** può assumere i valori l0, l1, l2.. l31 dove l0 corrisponde al primo colore impostato, l1 al secondo colore impostato...
  - **Y** definisce il valore di y per il quale il colore impostato e' riconosciuto.
  - **CrMin** e **crMax** impostano il minimo e massimo valore di cr, per il dato y, nel quale il colore e' riconosciuto. CbMin e cbMax hanno lo stesso significato per il campo cb.

  Il metodo setColor puo' essere invocato un numero arbitrario di volte per definire correttamente i valori di cr e cb per ogni possibile valore di y.
- `const unsigned int*` **`processImage`**`(unsigned char* pImage)`: elabora l'immagine passata come parametro e ritorna un puntatore const all'immagine elaborata.
  I pixel dell'immagine ritornata sono degli unsigned int, utilizzando i valori l0, l1... come maschere è possibile identificare se il dato pixel è stato riconosciuto o meno nel dato colore.
- `int` **`getNumberOfRecognizedPixel`**`(Colors selectedLayer)`: ritorna il numero di pixel trovati nel selectedLayer colore nell'ultima elaborazione effettuata.

Esempio di utilizzo

Di seguito un esempio di utilizzo di AISAiboYCrCbColorSeg utilizzato per il riconoscimento della AiboBall.

Verranno evidenziate solo le parti riguardanti l'utilizzo di AISAiboYCrCbColorSeg.

Nel **costruttore** è sufficiente creare l'oggetto AISAiboYCrCbColorSeg in modo da inizializzare le strutture per l'elaborazione dell'immagine. Al costruttore è necessario specificare le dimensioni delle immagini da elaborare.

```
//
// Costruttore di default
//
Occipital::Occipital() {

      ...

  // Creo il segmentatore. Devo specificare nel costruttore la dimensione
  // dell'immagine per permettere una migliore ottimizzazione dell'algoritmo.
```

```
  AISAiboYCrCbColorSeg segmentator(208, 160);

      ...

}
```

Nel metodo **DoInit** vengono inizializzati i colori che l'Aibo dev'essere in grado di riconoscere. Il metodo setColor ha una segnatura simile al metodo OcdtInfo::Set offerto dalle API di Open-R per il Color Detection, la differenza riguarda il valore di y. In Open-R y può assumere al massimo 32 valori, mentre in AISAiboYCrCbColorSeg y può assumere tutti i valori da 0 a 255.

```
//
// Inizializzo l'oggetto Open-R
//
OStatus Occipital::DoInit(const OSystemEvent& event)
{
    OSYSDEBUG(("Occipital::DoInit()\n"));

    NEW_ALL_SUBJECT_AND_OBSERVER;
    REGISTER_ALL_ENTRY;
    SET_ALL_READY_AND_NOTIFY_ENTRY;


   //
   // Apro la telecamera e imposto i parametri di Bilanciamento del bianco
   // e esposizione
    OpenPrimitives();

    SetCameraParameter();

   //
   // Imposto i colori:
   // in l0 il giallo
   // in l1 il rosa della Aiboball
   //
    for(int i=0; i< 256; i++)
      {
      // giallo
      seg.setColor(AISAiboYCrCbColorSeg::l0, i,  135, 165, 50, 98);

      // palla
      seg.setColor(AISAiboYCrCbColorSeg::l1, i, 150, 230, 120, 190);
      }


    return oSUCCESS;
}
```

L'entry point **NotifyImage** viene invocata al ricevimento di ogni nuova immagine. E' sufficiente passare il puntatore ritornato dal metodo ***GetData*** al metodo **AISAiboYCrCbColorSeg::*processImage*** per effettuare il riconoscimento dei colori. Una volta processata l'immagine ptr punterà ad una matrice di interi senza segno. Il metodo *getNumberOfRecognizedPixel* ritorna il numero di pixel riconosciuti per il dato colore ed è utile per effettuare un veloce controllo della presenza o meno dell'oggetto che si intende individuare, successivamente viene calcolato il centroide dell'oggetto rosa individuato.

```
//
// Metodo invocato al ricevimento di una nuova immagine.
```

```
//
void Occipital::NotifyImage(const ONotifyEvent& event){

 OFbkImageVectorData* imageVec =
reinterpret_cast<OFbkImageVectorData*>(const_cast<void*>(event.Data(0)));

  byte* data = imageVec->GetData(ofbkimageLAYER_H);

   //
   // Elaboro l'immagine in formato YCrCb Aibo RAW.
   //
  const unsigned char* ptr=segmentator.processImage( static_cast<unsigned
char*>(data) );

   //
   // Se ho trovato un BALL_THRESHOLD numero di pixel rosa allora l'ho trovata
   //
 if(segmentator.getNumberOfRecognizedPixel(AISAiboYCrCbColorSeg::l1) =>
BALL_THRESHOLD)
    {
      //
      // Palla trovata. Trovo il centroide
      //
      int xsum   = 0;
      int ysum    = 0;
      int npixels = 0;

     for (int y = 0; y < height; y++)
      {
        for (int x = 0; x < width; x++)
        {
        if (*ptr++ & AISAiboYCrCbColorSeg::l1)
          {
            xsum += x;
            ysum += y;
            npixels++;
          }
        }
      }
      int xcentroid  = xsum / npixels;
      int ycentroid  = ysum / npixels;
      ...
    }
...
}
```

Per compilare il codice, è sufficiente compilare AISAiboYCrCbColorSeg con il seguente comando:

```
g++ -ftracer -O2 -Wall -fomit-frame-pointer -o .AISAiboYCrCbColorSeg.o -c
AISAiboYCrCbColorSeg.cc
```

Quindi effettuare il linking dell'Oggetto Occipital includendo il file AISAiboYCrCbColorSeg.o.
L'opzione *-ftracer permette*, stando alla documentazione di gcc, di rendere più agevole l'ottimizzazione dell'oggetto, ottimizzazione che viene attuata con l'opzione *-O2*. *-fomit-frame-pointer* permette di velocizzare le chiamate a funzioni e rende disponibile un registro in più al processore migliorando l'ottimizzazione. Tutte queste funzioni permettono un notevole risparimio del tempo di calcolo, ma non sono necessarie.